Unity 3D

TP Prise en main et fonctions de base

Objectif :

Prendre en main l'interface Unity et manipuler ses concepts notamment à travers l'écriture de scripts C#
Utiliser la documentation officielle
Réaliser une petite application interactive 3D

Pré-requis

Lecture du cours de présentation Unity

0. Ressources

Logiciel : http://unity3d.com/unity/
Documentation et tutoriels : http://unity3d.com/learn

- Manual
- Scripting API
- Tutorials
- Resources

1. Construction du projet : interface & éléments de base

- Créer un nouveau projet (= dossier) (sans charger aucun package prédéfini)
- Dans l'onglet project, créer des dossiers Scripts, Materials, Prefabs et Scenes. Ils serviront à organiser vos assets.
- Enregistrer la scène courante (extension *.unity*) (NB : n'oubliez pas de sauvegarder régulièrement le projet **et** la scène !)
- Insérer un plan dans la scène
    - GameObject -> Create other -> Plane
    - Observer ses composants dans l'Inspector
    - Modifier sa position pour qu'il soit visible par la Main Camera (par les outils du menu et par les champs du composant Transform)
- Insérer un éclairage de type quelconque
    - GameObject -> Create other -> <type> light
    - Modifier sa position pour qu'il éclaire le plan
- Insérer et placer un cube
    - GameObject -> Create other -> Cube
- Créer un nouveau Material *Red*
    - Assets (ou clic droit dans le project) -> Create -> Material
    - Choisir la couleur rouge
    - Appliquer ce material au cube par glisser-déposer
- Utiliser une image pour texturer le plan
    - Enregistrer une image quelconque sur le PC
    - L'importer dans le projet : Assets -> Import new asset -> ... ou par glisser-déposer
    - L'ajouter au plan (notez la création automatique d'un material correspondant)

- Ajouter de la physique à la scène
    - Sélectionner le cube
    - Ajouter le Component –> Physics -> Rigidbody
    - Jouer la scène.
    - Quels components sont utilisés par le moteur pour ce comportement ? Modifier différents paramètres et tester
    - Pour la suite, on désactivera la gravité
- Glisser-déposer le cube depuis l'onglet Hierarchy vers l'onglet Project : cela va créer un « Prefab » à partir du cube (cf. icône bleue)
    - le renommer en CubePrefab
    - par l'opération inverse, insérer 2 autres cubes à partir de ce modèle dans la scène et modifier quelques paramètres

2. Programmation d'une première animation : rotation des cubes

- Créer un nouveau Script C# Rotate
    - Assets -> Create -> C# Script
    - L'éditer (par défaut avec Visual Studio)
    - Dans la fonction Update(), faire une rotation de 5° selon l'axe y (cf. cours et classe Transform) et sauvegarder
- Ajouter ce script au préfab CubePrefab par glisser-déposer dans l'onglet Project
    - notez l'apparition en tant que composant dans l'inspector ; on peut le désactiver ou le supprimer de l'objet (NB : Attention à ne pas l'ajouter plusieurs fois !)
    - notez également que toutes les instances de CubePrefab sont modifiées par cet ajout
- Tester
- Pour plus de généricité du script
    - Ajouter un attribut avant le Start() : public float speed = 5.0f;
    - Modifier la rotation précédente
    - Vous remarquerez l'apparition de cet attribut dans l'inspector pour pouvoir modifier sa valeur. Tester avec différentes valeurs.

3. Programmation d'interactions clavier simples

On souhaite contrôler l'activation de la rotation précédente :

1. Rotation tant que la touche R est appuyée
    - créer un C# Script RotateDuringKeyPressed
    - reprendre le contenu du script Rotate en ajoutant un test sur la touche R (cf. documentation Input)
    - confirmer ce test par un message dans la fenêtre de log (cf. documentation Debug ou print())
    - remplacer le script dans CubePrefab et tester (rq : désactiver la gravité)
2. Généralisation
    - Modifier le script précédent pour pouvoir choisir la touche de rotation dans l'inspector
    - Paramétrer 3 touches différentes pour les 3 instances de cube et tester
3. Rotation démarrée ou arrêtée par 1 seul appui sur 1 touche
    - créer un script RotateOnOffKey
    - quelle fonction de Input et avec quoi l'utiliser pour arrêter/reprendre la rotation ?
    - ajouter le script dans CubePrefab, désactiver le script précédent, paramétrer les touches dans les instances et tester

4. Programmation d'interactions souris simples

1. On souhaite changer la couleur du 1er cube lorsque la souris passe dessus :
   - Créer un script ChangeColorOnMouse
   - Pour la détection de l'interaction souris, utiliser des événements spéciaux des scripts (cf. MonoBehaviour)
   - Pour la modification de la couleur, 2 possibilités (cf. Renderer et Material) :
     - Modifier la couleur du matériau pour l'objet en cours
     - Ou changer directement le matériau utilisé
     - NB : La nouvelle couleur peut être créée/choisie simplement dans le script, mais pas le nouveau matériau qui doit être en attribut public et donné dans l'inspector
   - Pour retrouver la couleur ou le matériau original :
     - Où et comment les sauvegarder ?
     - Quel événement de script utiliser ?
   - Tester

5. Un peu de transformations spatiales

1. On souhaite naviguer en faisant orbiter la caméra autour et vers le centre de la scène avec les touches du clavier
   - Créer un script OrbitalCamera
   - Récupérer les valeurs des axes correspondants aux flèches verticales et horizontales
   - Appliquer ces valeurs dans les 2 rotations voulues (attention aux axes et repères de rotation)
   - Effectuer un changement de zoom avec + et -
2. On souhaite déplacer les cubes sur le plan selon les mouvements de la souris[1]
   - Créer un script ManipulationMouse
   - Dans Update
     - Tester l'appui sur le bouton gauche avec une fonction de Input[2]. Confirmer le test par un message dans la fenêtre de log
     - Stocker le déplacement souris entre chaque frame, toujours en utilisant une fonction de la classe Input
     - Effectuer la translation correspondante à ce déplacement souris
   - Ajouter au CubePrefab et tester
   - Proposez une solution pour que les cubes se déplacent uniquement si on a cliqué sur eux

6. Réutilisation de scripts d'interactions clavier/souris

- Télécharger les Standard Assets officiels sur l'Asset Store ou la page du cours
- Importer le package Character Controllers qui en fait partie
  - Assets -> Import Package -> ...
- Ajouter un First Person Controller (FPC) à la scène. Il possède sa propre main caméra : désactiver la main camera utilisée précédemment
- Modifier les positions du FPC, de la capsule et de la main camera pour avoir une vue à la 1ère personne, posée sur le plan

[1] Pour simplifier on considèrera que OrbitalCamera n'est pas actif lors de la manipulation

[2] Bouton gauche = 0 ou "Fire1" selon la fonction utilisée ; les détails des Inputs sont définis dans le projet : Edit -> Project settings -> Input

- Tester
- Etudier les scripts MouseLook et FPSInputController. Vous noterez en particulier les appels à Input.GetAxis().

7. Création dynamique d'objets

- Créer un nouveau Script CreateOnClick
- Ajouter ce script au First Person Controller
- Coder le script :
    - Créer 2 attributs pour référencer un modèle d'objet à copier et la copie
    - Tester si le bouton droit de la souris ("Fire2") a été cliqué
    - Instancier la copie à partir du modèle (avec Instantiate()) à la position/rotation courante (ie. celle donnée par le transform du propriétaire du script)
    - Changer la couleur de la copie en bleu
    - La positionner juste devant la position courante
    - Confirmer la création par un message dans la fenêtre de log
- Tester avec un prefab d'objet comme modèle

8. Build

- Construisez votre scène de la manière suivante pour pouvoir tester vos scripts
    - 3 cubes avec chacun une méthode de rotation différente (normale, during key pressed avec R, onoff avec T)
    - Chacun avec une couleur différente lors du passage de la souris
    - Chacun manipulables en translation à la souris
    - Une caméra orbitale (désactiver le FPS)
    - La création dynamique d'une sphère
    - Ajouter un script qui quitte l'application lors de l'appui sur la touche Echap
- Générer votre application pour Windows
- Observez les fichiers/dossiers générés (variables selon la version de Unity)
- Tester l'exécutable
- Zipper le dossier complet de l'exécutable

9. Enrichissement de la scène

Explorer les Assets, les GameObjects, les Components et la documentation pour ajouter des propriétés ou des animations à votre scène (eg. particules, sons...).

Annexes : quelques conseils d'utilisation et raccourcis

Sauvegarder (control + s) très régulièrement le projet **et** la scène
Les changements de la vue dans la scène vous permettent de mieux visualiser les 3 dimensions et les positions relatives de vos objets et caméras : - translations = clic milieu - rotations = clic droit ou alt + clic gauche - zoom = molette - zoom auto sur objet sélectionné dans la hiérarchie = Shift + F
Lancer et arrêter le jeu avec le bouton Play Eviter la mise en pause : toutes les modifications que vous faites dans l'inspector pendant la pause seront perdues lors de l'arrêt (qui revient aux conditions initiales)
Si vos objets ne sont plus visibles : revenir à des positions/orientations canoniques via l'inspector (par ex. (0,0,0)) pour ces objets ET pour la caméra
Privilégiez une disposition des vues qui ressemble à celle du cours : onglets scène et game visibles en même temps, console visible… Si vous perdez/fermez des onglets ou que la disposition ne vous convient pas : Menu Window ou Window -> Layout Dans le pire des cas, sauvegardez et fermez l'onglet concerné par le problème, voire fermez le projet. Au démarrage d'un projet aucune scène n'est chargée (donc hiérarchie et scène vide), il faut la lancer manuellement
Nommez de manière explicite vos objets, prefab, scripts, matériaux et rangez les dans des dossiers également bien nommés
Supprimer un composant entraine la perte des paramètres si vous les avez modifiés. Préférer une désactivation (case à cocher) pour les tests si c'est possible